Application of Genetic Algorithms to a Multi-Agent Autonomous Pilot for Motorcycles

Poornima Konanur
Department of Computer Science
Indiana University of South Bend
E-mail Address: npoornima@iusb.edu
Date: May 10th 2005

Thesis submitted to the faculty of the
Indiana University South Bend
in partial fulfillment of requirements for the degree of

MASTER OF SCIENCE

in

APPLIED MATHEMATICS & COMPUTER SCIENCE

Advisor

Dr. Dana Vrajitoru
Department of Computer Science

Committee:
Dr. James Wolfer
Dr. Morteza Shafii-Mousavi

Abstract

The Physics behind motorcycle driving are well understood and implemented by studying the laws of kinetics and kinematics behind the operation of the single track motor vehicle.

In this thesis I worked with an application which is currently using OpenGL and implements an interactive motorcycle simulator which is based on the laws of physics. This application involves a multi-agent pilot capable of autonomously driving the vehicle using some configurable equations.

I have applied genetic algorithms to find suitable values for the parameters of the pilot by testing it in a non graphical environment, and I visually verified the results of the genetic algorithms with the graphical interface application. The performance of the pilot derived by the genetic algorithms is also compared with the manually configured pilot.

Acknowledgements

First of all I wish to express my sincere gratitude to my advisor Dr. Vrajtoru during this work. This work would not have been possible without support and encouragement of Dr. Vrajtoru, under whose supervision I chose this topic and began thesis.

I would like to acknowledge the help of Dr. Wolfer and Dr. Shafii Mousavi for their support and guidance during the thesis. I would like to thank Dr. Hakimzadeh for help with the experimental setup and general advice.

Finally, I would like to dedicate this thesis to my parents, my husband Subbu, and daughter Raksha for their love, patience, and understanding; they allowed me to spend most of the time on this thesis for past couple of months.

Table of Contents

List of Tables

List of Figures

1. Introduction

Many real - world and scientific applications make use of autonomous techniques, like autonomous robots (Al-Shibabi, Mourant, 2003; Sukthankar, Baluja, Hancock, 1998), automated flight control (Abdelzaher, Atkins, and Shin, 2000; Atkins, Miller, VanPelt, Shaw, Ribbens, Washabaugh, and Bernstein, 1998; Gavrilets, Frazzoli, Mettler, Piedmonte, and Feron, 2001), traffic control (Kelly, 1997; Nagel, 1996; De Mot , Feron, 2003; Mourant , Marangos, 2003).

The intelligent agents represent a modern approach in artificial intelligence and they have been extensively utilized for many applications. Several approaches have applied multi-agent models to the simulation of autonomous drivers and this application follows similar ideas. A related research direction focuses on traffic flow simulation or trajectory planning.

The paper that motivated my interest is (Vrajitoru, Mehler, 2004) in which they describe a motorcycle simulation implemented using the OpenGL library and providing real time interaction for a human player. They developed a visual interface that allowed a human user to change the point of view and drive the vehicle which in this case is a motorcycle. This application currently includes an autonomous pilot that can be toggled on and off and also a test circuit that a human or an automatic driver must attempt to complete. The autonomous pilot is a multi-agent probabilistic application with a separate configuration interface where each process is acting on one of the control units of the vehicle like gas, brakes, the handlebars (equivalent to the steering wheel for a car). The agents use some of the information about the current status of the vehicle to make a decision about an action to be taken on the control units they are in charge of. The information includes both status data, like the current speed, and perceptual data, like the visible distance on the road in the direction of movement, the lateral distance to the border of the road and the current slope.

The genetic algorithms (Holland 1975; Goldberg, 1989) are widely used learning and optimization method with many applications especially fields related to our current research, in particular to robotics (Sedighi, Ashenayi, Manikas, Wainwright, Tai, 2004).

My goal in this project was to apply genetic algorithms (GAs) to configure the agents composing the autonomous pilot. Thus, each agent has a behavior determined by a set of equations involving some coefficients and thresholds, as I shall briefly describe in Section 3. All of these are currently configured by a human player by trial and error, which is a time consuming and imprecise process.

By applying the GAs to find the optimal values for all of the coefficients involved in the equations of the agents, the process of deriving a good behavior for the autonomous pilot can be made automatic. In the same time I achieved a better performance than it is possible by simply adjusting these parameters by hand.

To better explain the research topic, Section 2 presents the literature survey of the bicycle dynamics and other related topics. Section 3 introduces the physical model of the motorcycle followed by the control units of the motorcycle. Section 4 explains one major contribution of this thesis which is the automatic testing of the circuit completion and detection of various crash and failure conditions. Section 5 presents the implementation of the genetic algorithms to apply them to the current problem. Section 6 introduces the experimental results and a comparative study of the GA with the previous pilot. Finally, Section 7 presents some conclusions.

2. Literature Review

The need for autonomy for vehicles arises from the difficulty of always having a human operator controlling the system. The interest in multi agent autonomous pilots has been rapidly increasing in last few years. In a research paper (Mohan, Busquets, Lopez de Mántaras, Sierra, 2004) considers a multi-agent pilot which in this case operates a robot capable of navigating in inaccessible environments which are usually unknown and unstructured (Mars, Moon). The pilot functions as an autonomous agent in a complex multi-agent architecture for the control and navigation of an autonomous robot. In this architecture, various agents are responsible for different tasks, and they might have to compete and cooperate for a successful completion of a particular navigation mission. They proposed a general architecture that uses a bidding mechanism to control the robot. In this case, they used it to coordinate the three systems that control the robot, which are navigation, vision, and pilot.

The first project in the field of bicycle dynamics (Olsen and Papadopoulos, 1988) was created to apply modern scientific techniques to the engineering problems of the bicycle. They applied the mathematics to include the important aspects of geometry and mass replacement. They selected to work on a basic bicycle model that had rigid knife edged wheels, a rigid rear frame including a rigidly mounted and immobile rider, and a rigid steer able front fork, including front wheel, stem, and handlebar. The equations they obtained were not as straightforward as mass or spring fitness, but rather involved functions of bicycle velocity, frame geometry, and various characteristics of the bicycle and rider's distribution of mass and also the leaning affects on steering and vice versa.

Another project taken by Cornell Bicycle Research Project (Fuchs, 1998) considered minimizing the disturbing effects of steady crosswinds on single-track vehicles. The equation to calculate the equilibrium location of the center of pressure for zero steering angles in crosswinds -

the ‘trim equation’ - has been derived. Using it, a single-track velomobile designer may trim their vehicle to achieve good handling characteristics under certain conditions (angle of attack); the torque that has to be exerted by the rider onto the handlebar may be minimized. But the fact that a vehicle is in trim at certain angles of attack does not assure safe handling in any situation that may be encountered in windy conditions on the street. For the first time it was mathematically shown that static stability of single-track vehicles in crosswinds is achieved when the center of pressure is in front of the center of mass. In this research they have not considered the dynamics of transition from one state of crosswind influence to another state of cross wind influence.

Fajans (2000) considered centrifugal forces that will throw the bike over the side if the rider steers the handlebars in the opposite direction of the desired turn without first leaning the bike in to the turn. Leaning the bike into the turn allows for the gravitational forces to balance the centrifugal forces, leading to a controlled and stable turn. Thus steering a bike involves a complicated interaction between centrifugal and gravitational forces, and torques applied to the handlebars, all edited by the bike geometry.

The genetic algorithms have been applied to local obstacle avoidance of a mobile robot in a given search space (Sedighi, Ashenayi, Manikas, Wainwright, Tai, 2004). In this research they have tried to derive not only a valid path but also an optimal one. The objectives were to minimize the length of the path and the number of turns taken by the robot to complete the path. They have also implemented a method that allows a free movement of the robot in any direction so that the path planner can handle complicated search spaces.

The research for this thesis is a continuation of the project presented in (Vrajitoru, Mehler, 2004). Starting from the existing model for the motorcycle and the pilot, we apply genetic algorithms to configure the autonomous pilot.

This paper introduces an application that simulates a motorcycle that can be driven by both a human player and an autonomous pilot. The application is implemented based on the physical equations describing the vehicle's attributes, motion, and road behavior. This application aims at controlling the vehicle in a non deterministic way inspired from the behavior of a human driver and using similar perceptual information to make decisions.

The goal of this application is to simulate the behavior of a human driver under various circumstances on the road.

The application presented in this paper (Vrajitoru, Mehler, 2004) is developed using the ideas and concepts that can be observed in game engines. It is implemented using the OpenGL library and provides real-time interaction for a human player. The visual interface of the application allows the human user to adjust the point of view and to drive the vehicle, which in this case is a motorcycle. The application includes an autonomous pilot that can be toggled on and off as well as a test circuit that the human or autonomous driver must attempt to complete. The autonomous pilot is a multi-agent probabilistic application with a separate configuration interface where each agent is an independent process acting on one of the control units of the vehicle, as for example, the gas, the brakes, the handlebars, or the steering wheel. The agents use some information about the current status of the vehicle to make a decision about an action to be taken on their respective control units. This information includes both status data, like the current speed, and perceptual data, as the visible distance on the road in the direction of movement, the lateral distance to the border of the road, and the current slope. The performance of the autonomous pilot is compared with the performance of a trained human.

3. Motorcycle Modeling and Autonomous Pilot

In this section we introduce the physical model of the motorcycle and the multi-agent autonomous pilot.

3.1 The Vehicle Control and Degrees of Freedom

The motorcycle is a single track vehicle (STV) modeled as a system of several units with various degrees of freedom as shown in Figure 1. The coordinate system relative to the motorcycle is also illustrated. The origin of the system is in the center of the motorcycle at the ground level.

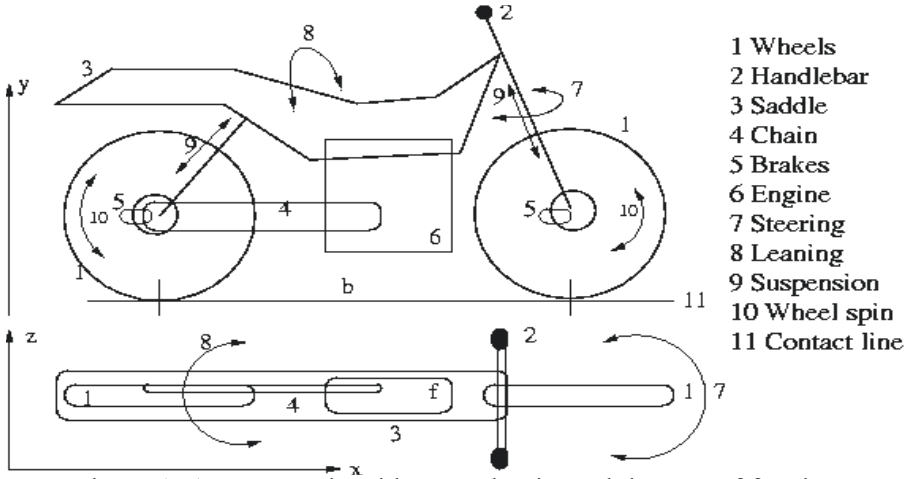

Figure 1. A motorcycle with control units and degrees of freedom

The motorcycle has six degrees of freedom namely:

- Rotation of the wheels around their local axes which are parallel to Oz.
- The rotation of the handle bar and the front wheel around the fork axis (steering).
- The front and back translation along the suspension axes. These components are not taken into account by our system.
- The rotation of whole vehicle around the Ox axis in the relative system of coordinates (leaning).

The driver can control the vehicle with five the control units which are the handlebar (steering), leaning of the vehicle laterally, the throttle, and also the two brakes namely, front and rear. The state of the STV is described at any moment by the current position of the center of the vehicle on the road and the current direction of movement which can be described as a vector or as an angle in the (x,z) plane plus the slope, which is in general determined by the road. The model must also include the degrees of freedom for each of the control units. These components are in general defined relative to the STV's internal system of reference.

3.2 STV Motion and Control

The STV is modeled as a reduced state system of continuous variables. The generalized coordinates at a particular moment are given by

$$q = (s, a, ?)^T \quad (1)$$

where $s(t) = (x(t),z(t))$ represents the spatial position of the STV,

$?$ is orientation angle determining the direction of movement $d = (\cos ?, \sin ?)$,

a is the leaning angle.

The state of the vehicle also involves F, the steering angle. The constraint imposed on this angle is $-p/3 = F = p/3$. The Figure 2 shows these angles and coordinates.

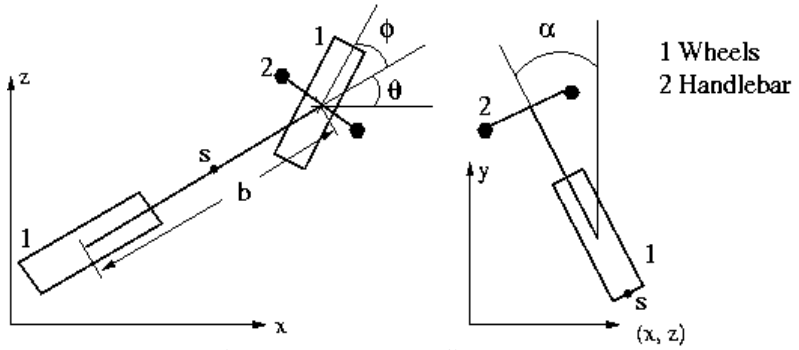

Figure 2. STV coordinate system

The vertical components of both s and d are determined by the road altitude and slope at the given spatial position and also by the vehicle orientation. In this thesis I have considered the

road to be close enough to the sea level such that the gravitational acceleration is the constant $g = 9.8m/s^2$ and the altitude of the vehicle has no noticeable effect on the motion of the vehicle.

Let *s (s, d)* be the angle made by the contact line of the vehicle with the *(x, z)* plane, as determined by its position and orientation.

The driver's input to the system can be represented as follows.

$$u = (t, \beta_f, \beta_r, F, a) \tag{2}$$

where t is the throttle opening determining the acceleration in the direction of movement d.

β_f, β_r are the front and rear brakes respectively, and

F and a are the steering and leaning angles respectively.

A nonholonomic system is one whose movement at any moment cannot instantly change direction, but is subject to certain constraints. In our case, the nonholonomic constraints arise from the fact that the motorcycle moves in the direction of wheels which is perpendicular to their main axis. A change in direction is not instantaneous, but determined by some actions (steering and leaning). This change has a limited range given by the maximum degree of freedom of the handlebar and maximum leaning that doesn't cause the vehicle to fall.

The nonholonomic constraints can be expressed by following, where b is the distance between the two contact points of the wheels on the ground.

$$-x'\sin? + z'\cos? = 0 \tag{3}$$

$$b\cos F\ ?' - \sin(F+?)x' + \cos(F+?)z' = 0 \tag{4}$$

where the single and double quotes are the standard notations for the first and second derivatives respectively with respect to the time.

Equation 3 represents the fact that the STV moves in the direction of the vector *d*. Equation 4 allows us to compute the change in orientation due to steering. In particular, if –p/2 = *F* = p/2, (heuristic) we can compute the change in the orientation angle due to steering as

$$? ? = (\sin (F+?) ? x - \cos (F+?? z))/b \cos F \quad (5)$$

Let v = s' be the momentary speed or velocity in the direction of movement, and a = v' = s? the momentary acceleration in the direction of movement. We implemented the motion of the vehicle using Newtonian mechanics.

Let us consider the following notations:

_ $s(t)$ the spatial position of the object at time t,

_ $v(t)$ the momentary speed or velocity, $v(t) = s'(t)$,

_ $a(t)$ the momentary acceleration, $a(t) = v''(t) = s?(t)$.

By applying Newton's laws of motion, we can derive the following system of equations to describe the spatial position of the motorcycle at the moment $t + ? t$ by

$$s(t + ? t) = s(t) + ? s \quad (6)$$

$$v(t + ? t) = v(t) + ? v \quad (7)$$

$$? s = d (v ? t + a ? t^2 / 2) \quad (8)$$

$$? v = a ? t$$

In our case, the acceleration is defined by the throttle which determines the amount of fuel supplied to the engine, by the force applied to the brakes, by the friction force, and by the gravitational force when the road is not flat. The system is set in such a way that a given amount of gas supplied to the engine can only lead to accelerating the vehicle up to a speed limit depending on the amount of gas. This simulates the engine limitations of a real vehicle. The brakes do not act simply as a negative acceleration but also have the effect of adding to the friction and drag forces which otherwise just depend on the air and ground and are relatively very small.

$$a = t + g \sin s - k\, g \cos s - k_b(B_f + B_r) - D\, v^2 \quad (9)$$

In this above equation,

g = 9.8 m/s^2 is the gravitational acceleration at the sea level,

k = coefficient of friction,

k_b = coefficient of friction for the brakes,

D = coefficient of drag, defined as the sum of the air resistance, the force applied to the brakes, and the engine brake, as follows

$$D = k_a + k_d(B_f + B_r) + k_e \quad (10)$$

where k_a is the air resistance, k_d is the coefficient of drag for the brakes, and k_e represents the engine brake.

These forces mentioned above cause the speed of the vehicle to become constant after a while for any given throttle opening t as long as the road conditions are stable. Along with the friction force, they prevent a resting vehicle from going downhill if the slope s is not null, and prevent the speed from increasing indefinitely due to gravitation in the direction of movement when the vehicle is going downhill.

In this model of the motorcycle we also consider that when the motorcycle leans more than a threshold, the centrifuge force cannot compensate for the gravitation anymore and the vehicle falls down (crashes). The threshold depends on the speed, a higher speed allowing the vehicle to lean further without crashing.

Leaning Equations

The first force that we are going to consider that affects the change in direction of the vehicle due to leaning is the centrifuge force. This force is defined by

$$F_c = m? \,^2 r \quad (11)$$

where $?$ is the angular speed, and r is the radius of the circle on which the object is turning. If v is the horizontal speed, then we can define the angular speed as $? = v/r$, so the centrifuge force is equal to

$$F_c \quad m\frac{v^2}{r} \qquad (12)$$

A second force that interacts with the vehicle in the lateral movement is the lift due to friction with the air. We can adapt an equation taken from airplane wing simulation that computes the *lifting force* F_L is given as

$$F_L \quad \frac{1}{2}\rho\, v^2 S_{ref} C_L \qquad (13)$$

In this equation ρ is the air density, that we can consider to be approximately ρ = 1.22145kg/m^3 at 0 altitude.S_{ref} is the reference area, that we can compute as the horizontal projection of the vehicle. If S_v is the total porting lateral surface of the motorcycle and the driver, S_h the porting horizontal surface of the motorcycle, and a is the angle made by the vertical axis of the motorcycle with the horizontal plane, then

$$S_{ref} = S_v \cos a + S_h \sin a \qquad (14)$$

The last component of the lateral movement is the gravitational force itself, which has a norm equal to $g\ m$. From this force, we have to subtract the lifting force first. Starting from the same angle a, the resulting gravitation force which is vertical is decomposed in a force along the vertical axis of the motorcycle and another one that is normal to the motorcycle. The rotation will be determined by the component that is perpendicular to the motorcycle axis. This component, that we call central gravitation and denote by G_c, is given by

$$G_c = (g\, m - F_L) \sin a \qquad (15)$$

By imposing the condition that the central gravitational force should be equal to the centrifuge force, we can compute the rotation radius r:

$$r \quad \frac{mv^2}{(g\,m - F_L)\sin\alpha} \tag{16}$$

All of the forces and quantities involved in the description of the lateral movement are illustrated in Figure 3.

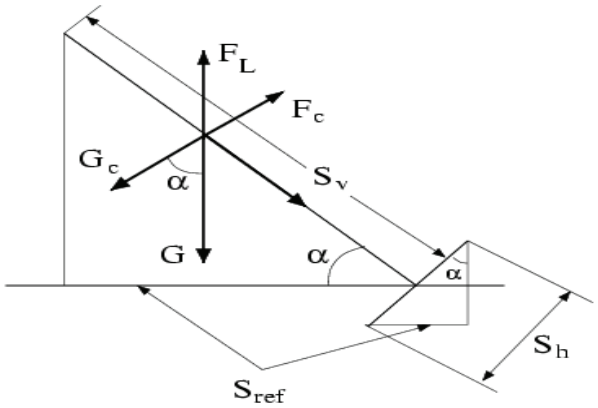

Figure 3. Forces and quantities involved in lateral movement

3.3 Perceptual Information

The autonomous pilot that is developed in this application is actually using perceptual information to make decisions in regard to the vehicle driving. This is done to simulate the perceptual cues from real life that a human driver pays attention to while driving the vehicle. In this application the cues given to the pilot are the following:

The *visible front distance*, denoted by *front,* is defined as the distance from the current position of the vehicle in the direction of movement to the border of the road scaled by the length of the vehicle. This information tells us how much of the road is visible to the driver and also how straight the road is in front of the vehicle. We will also refer to this measure as the *horizon*.

The *front probes*, denoted by *frontl* and *frontr,* are defined as the distances to the border of the road from the current position of the vehicle in directions rotated left and right by a small angle from the direction of the movement of the vehicle. This gives the pilot information

regarding on which side of the current direction of motion the front distance would be greater, indicating which direction the road turns.

The *lateral distances,* denoted by *leftd* and *rightd,* define measures of the lateral distances from the vehicle to the border of the road, at a short distance in front of it, simulating what the pilot might be aware of without turning their head to look. A high value of this measure indicates a turn in the road or that the vehicle is close to the border. The value of this measure close to 0 indicates that the vehicle is in the center of the road.

The slope denoted by a *slope*, is a perceptual version of s which is discretized to simulate the intuitive notion of the road inclination that a human driver would have, as for example almost flat, slightly inclined up or down. This simulates the fact that the pilot is not aware of the precise value of s. The Figure 4 represents the geometrical definition of the measures defined above.

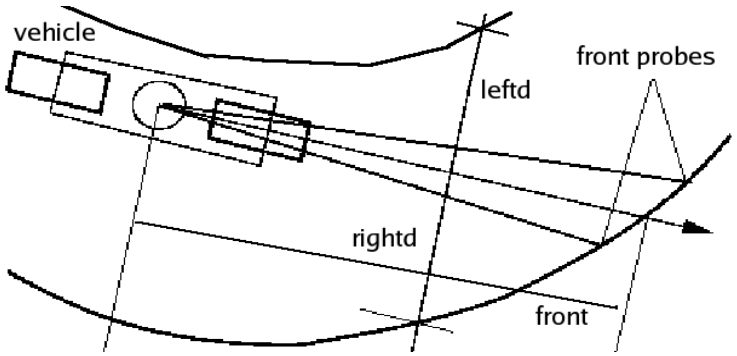

Figure 4. Perceptual information used by the autonomous pilot

Apart from the perceptual information, the autonomous pilot uses the current status of the motorcycle to make decisions about the actions to be taken on each of the control units. The status includes measures like the current speed, the current opening of the throttle, the brakes, and current deviation of the handlebar from the direction of the movement. These values can be expressed as a tuple (v, r, B_f, B_r, F).

3.4 Multi-Agent Pilot

The autonomous pilot that was the starting point for this thesis is composed of several agents. The model is based on the fact that the motorcycle can be driven using several control units (CUs). Each of them is controlled by an independent agent with a probabilistic behavior. The agents are not active during the computation of every new frame simulating the evolution of the vehicle on the road, but only once in a while in a non-deterministic manner. This simulates the behavior of a human driver that may not be able to instantly adapt and take action based on the road situation and would require a certain reaction time. The minimal model requires a CU for the gas - throttle, which determines the acceleration, for the brakes, which can slow down or even stop the vehicle, and for the handlebar that controls the direction.

Each of these control units is independently adjusted by an agent. The behavior of the agents depends on the status of the vehicle and is intended to drive the motorcycle safely in the middle of the road and at a safe speed as close as possible to a given limit. Each agent can have its own rate of interference with the coordination of the vehicle, and in our case, the agents controlling the throttle and the handlebar are in general more active than the agent controlling the brakes.

Next we will introduce the equations used by each of agents to make a decision and perform an action.

The Throttle

This control unit is responsible for controlling the opening of the throttle which determines the amount of fuel supplied to the engine and implicitly influences the speed of the vehicle. The input given to this agent can be represented by (*v; front; leftd; rightd; slope*). The agent uses a minimal speed threshold v_{low}, a maximal speed threshold over which the speed is considered unsafe, and the given speed limit v_{limit}. The agent aims to keep the vehicle speed above

v_{low} and below the maximal one, and also close below the v_{limit} which is an external measure that does not depend on the configuration of the driver.

The agent will determine the action to be taken based on the following considerations. If the lateral distance to the left is too different from the lateral distance to the right, the speed must be decreased because the road is most likely turning. The same rule applies to the visible distance in front of the driver: a short distance represents an unsafe road situation and the speed has to be decreased.

Let t be the throttle opening at the moment t, which in turn determines the acceleration of the vehicle. Let us also denote by lat_{norm} the normalized difference between the left and right distances as shown in Equation 17 and by $lat_{abs} = |lat_{norm}|$ the absolute value of this quantity.

$$lat_{norm} \quad \frac{leftd - rightd}{\max(leftd, rightd)} \tag{17}$$

Equation 18 presents the condition that must be fulfilled for the throttle to be increased or opened, which results in a higher acceleration followed by an incrementation of the speed. In this equation, v_{low} is a lower limit for the speed, thr_{lat} is a threshold under which we consider that the difference between the left and right distances is still safe, thr_{front} is the threshold for the safe front distance in front of the vehicle v_{limit} is the upper speed limitation, like the legal speed limit on that road, and c_{tr} is a constant.

$$\begin{cases} v(t) <= v_{low} & \text{or} \\ lat_{abs} > thr_{lat} & \text{and} \\ front >= thr_{front} & \text{and} \\ v(t) <= v_{limit} & \text{and} \\ v(t) <= c_t\, tr(t) \end{cases} \tag{18}$$

Let us denote by tr_{lat} a quantity indicating if the normalized absolute difference between the left and right distances is safe for the vehicle's current speed, as shown in Equation 19, where c_{vlat} and $pvlat$ are constants. For higher values of the speed, the safe difference is smaller.

$$tr_{lat} = lat_{abs} - (_{Cvlat)/}(1+v(t))^{1/pvlat}) \quad (19)$$

Let us denote by tr_{fr} a quantity indicating if the front distance is safe for the vehicle's current speed, as shown in Equation 20 where c_{vfr} and $pvfr$ are constants.

$$tr_{fr} = (c_{vfr/}(1+v(t))^{1/pvfr}) - front \quad (20)$$

Equation 21 represents the condition to be fulfilled for the throttle to be decreased or closed, which will have the effect of slowing down the vehicle under the influence of the friction force.

$$v(t) > v_{limit} \text{ and } tr_{lat} > 0 \text{ and } tr_{fr} > 0 \quad (21)$$

Let us denote by $?t = t(t + ?t) - t(t)$. The equation governing the change in throttle that the agent will perform based on the current vehicle and road status is illustrated by Equation 22, where c_{incv}, c_{decv}, and c_{sl} are constants. The actual amount of the change is a probabilistic quantity equally distributed in a small neighborhood around the computed value.

$$?t = c_{incv}(front - thr_{front})(v(t) - v_{low}) + c_{decv}((v(t) - v_{limit}) + t_{rlat} + tr_{fr}) + c_{sl}\, slope \quad (22)$$

The actual $?t$ which will be applied is given by $?t_{actual} = \text{randouble}(0.8,1.2) ?t$, where randouble is a function generating a real number uniformly in a given interval. This simulates the imprecision of a human driver and has no other physical meaning. For this reason we considered that the uniform distribution was sufficient.

The Brakes

The brakes agent is using the same idea as we have seen for the throttle agent as it is assumed that all the rules which decide whether the speed should be reduced or increased are of

general purpose and apply to all of the agents that have an influence on the speed. The equations for this agent are simpler as the brakes can only reduce the speed at any time.

Even though the constraints used in the equations for the brakes are the same for the throttle, the coefficients in these equations can have different values than those used for the throttle. As we know, the brakes are activated less frequently than the throttle when a human is driving a motor vehicle, because there is a seldom a need for such a drastic decrease in the speed that closing the throttle is not enough.

Let us say that $B_{r;f}$ is the amount of force applied to the brakes at time t. In our assumptions the force is distributed 60% on the front brakes and 40% on the back brakes. The brakes are handled by Equation 23 and have a probabilistic behavior similar to the throttle agent , but we can adjust the constants and thresholds independently of the agent controlling the throttle.

$$? B_{r;f} = c_{decv} ((v - v_{limit}) + tr_{lat} + tr_{fr}) - c_{slope} \quad (23)$$

From this equation we can see that a force is applied to the brakes if the speed is higher than the limit, if the front distance is too small, or if the vehicle is much closer to one lateral side of the road than to the other.

The Handlebar

This agent is in charge of controlling the handlebar of the motorcycle and determines if the rotation should be applied to the handlebar at a given moment, and what should the rotation angle be. This is the equivalent of the steering wheel for a car when it has to make a turn. This agent is using the lateral distances to the border of the road that are *leftd* and *rightd,* and also *frontl* and *frontd* which are the front probes. The agent turns the handlebar in the direction of the longer distance between *frontd* and *frontl* which has the effect of getting away from the closest border. The agent first considers the distance to either side, given by lateral distances. Thus, if the

vehicle is not within a given percentage (20%), of the center of the road then the agent moves the handlebar to direct the vehicle towards the center of the road. =

=

If the first measure does not provide the conditions to make a turn, the agent estimates the = distance forward to =the horizon (or front distance). Based on =the front probes and =the front = distance, the agent moves towards the center of the horizon. The angle by which the handlebar = turns depends on the distance to the horizon: the angle is bigger if the horizon is closer. =

=

The agent decides whether to use the lateral distances as =reference or the front probes = based on criteria shown in equation 24. Let us =denote by $probe_n$ the normalized =difference = between the front and right probes as in the equation below and by =$probe_{abs}$ = $|probe_{norm}|$ the = absolute quantity. =

$$probe_n \quad \frac{frontl - frontd}{\max(=frontl,=frontd)} \tag{24}$$

Let us denote by =lat_{diff} the quantity used by agent to decide if it must turn in which = direction as given by the equation below. =

$$lat_{diff} = \begin{cases} lat_{norm} & if\ lat_n > thr_{3=} and\ front_d > thr_{4=} \\ \frac{lat_n + probe_n}{2} & if\ latn > thr_{3=} and\ front_d >= thr_{4=} \\ probe_n & otherwise \end{cases} \tag{25}$$

=

where thr_i, i = 1, 4 are configurable coefficients. =

The handlebar agent will update the position of the handlebar if the condition expressed = in Equation 26 is fulfilled. This means that =the change is necessary if the lateral difference = measure is greater than the threshold thr_{lat}, or if the distance in the direction of movement to the = border is smaller than another threshold thr_{front}. =

Let us denote $?F = F(t + ?t) - F(t)$ the change in the handlebar angle decided by the agent. Then the general rule for modifying the rotation of the handlebar is shown in Equation 27. The actual amount of the change is a probabilistic quantity equally distributed in a small neighborhood around the computed value as in the following equation: $? F_{actual}$ = randouble (0.8, 1.2)* $? F$.

$$? F = c_{hbar}(lat_{diff} + (thr_{fr} - front)/ thr_f) \quad (27)$$

The amount of change in the direction of the handlebar depends on how different the left and the right lateral distances are either right next to the vehicle or at the intersection of the road in front of it, based on the measure of lat_{diff} and on the speed. If the motorcycle is at a lower speed the handlebar has to be turned more to achieve a given change in the direction. If the vehicle is moving at a higher speed then a small change in the orientation of the handle bar will obtain the same change in the direction.

Leaning Agent

The following which is in charge of controlling the leaning of the motorcycle. This agent also uses the lateral distances to the border of the road that are *leftd* and *rightd* and also *frontl* and *frontd* which are the front probes as used by the handlebar agent. The agent leans the vehicle in the direction of the longer distance between *frontd* and *frontl* which has the effect of getting away from the closest border. The agent first considers the distance to either side, given by the lateral distances. This agent measures the distance forward to the horizon. If the motorcycle is not within the given percentage (20%) of the center of the road, then the agent checks whether the motorcycle in on left side of the road or on the right side of the road and it leans the vehicle towards the center of the road. If the distance to the horizon is smaller then it checks the whether the curve of the road is to the left or to the right and it leans accordingly. A condition is also checked to see if the motorcycle if already in left or right leaning mode. In that case the agent cannot lean again in the same mode.

Alerting Agent

Another agent which is important for the functionality of the motorcycle is the alerting agent which does not have any direct control on the vehicle , but interacts with other agents. While other agents are active occasionally, this agent is probing the vehicle and the road condition for every new frame and is capable of activating one of the other agents if the situation requires extra attention. That is if the speed of the vehicle is too slow or too fast, or if the visible front distance is too short, or if the difference between the left and right lateral distances is too high, then this agent considers the situation as not safe. In this case, the agent generates an event which randomly activates one of the agents that can take action and correct the issue.

The following equations describe the conditions that must be true for the alerting agent to consider that the state of the vehicle is unsafe and trigger one of the agents coordinating the vehicle to take some action and correct the situation. The alerting agent will only generate an alert message but does not decide which other agent will perform the necessary action.

$$\begin{cases} v < c_{vlow} v_{limit} & or \\ v > c_{vhigh} v_{limit} & or \\ lat_{abs} > thr_{lat} & or \\ front < thr_{fr} \end{cases} \tag{28}$$

4. Pilot Testing and Evaluation

In this section we discuss application details from the previous project and also some of the improvements done.

4.1 Application Details from the Previous Project

The application is capable of driving the motorcycle by receiving input from both a human player and an autonomous pilot. The circuit consists of 3 loops as in Figure 4 a portion of the road is elevated with respect to the rest of the road. The circuit is designed such that it will

test the ability of the pilot when road turns both left and right where the slope of the road is ascending and descending.

The autonomous pilot was previously configured by hand and was capable of completing the circuit with some average speed considerably slower than the human player. The speed of the autonomous pilot showed an interesting behavior compared to the human player. In the case of the human player, the entire set of keys was hardly used and after the player is comfortable with speed of the vehicle they can complete the circuit by just using the lateral movement of the motorcycle. The pattern of variation in speed makes the simulation very close to the real life. In case of the autonomous pilot we can observe that autonomous pilot was more sensitive to the difference between left and the right distances to the border of the road. The number of turns taken is by the pilot was higher more compared to human driver.

4.2 Testing for Completion of a Circuit

The Figure 5 shows the perspective view of the circuit that has been used.

Figure 5. The test circuit

The Figure 6 shows the main window of the application displaying the motorcycle and the road with some perceptual cues, the outline of the road triangulation, and also the centerline. There is also a sub window containing a mini-map of the circuit which shows the position of the motorcycle. This helps the human player to locate the position of the vehicle on the entire circuit and lets them know how much of the circuit has been completed.

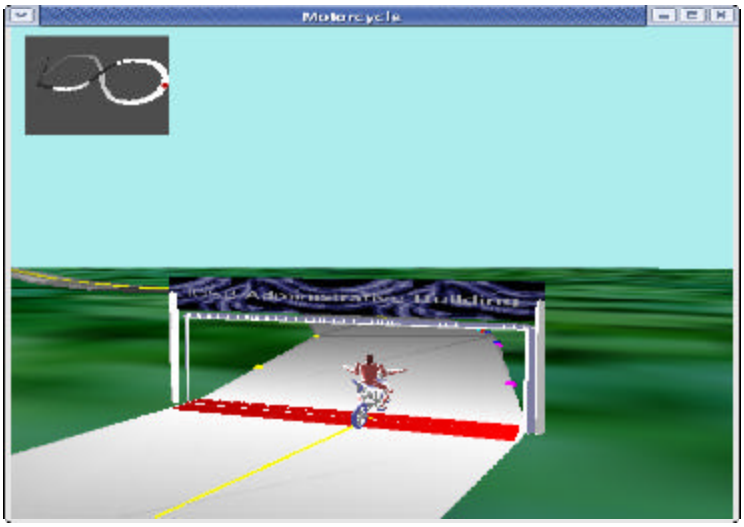

Figure 6. The main application window displaying the vehicle

In order to improve the GUI, I used texture mapping in OpenGL library to add some banners to the scene. One of them can be observed in Figure 6. This is very helpful for tracking the position of the vehicle.

In the previous version of this application there was no functionality to check the completion of the circuit. So the user had to toggle the pilot off after they observed that the motorcycle did complete a circuit. Part of my work has focused on checking for the completion of a circuit. For implementing this I considered a centerline which was drawn on this road. Since the motorcycle starts from the initial position I calculated the minimum distance between the position of the motorcycle and the centerline coordinates. After the motorcycle starts its motion, at every frame I check which of the centerline points the motorcycle is the closest to. A counter is keeping track of how many times the vehicle is closest to each point of the centerline. Each time I check whether the current position is the closest to the same centerline point as before or if it is nearer to the previous point (in this case the motorcycle is going backwards) or to the front point (in this case the motorcycle is moving forward). In a perfect case, the motorcycle will pass next to all of the centerline points once. The value of the counter for a point being greater than 0 means that the

vehicle was closest to that point at least once. These counters will show the behavior of the vehicle and can be used to test the completion of the circuit.

4.3 Testing for Different Crashing Conditions

There are four possible cases in which the test of one circuit ends when the motorcycle runs in the pilot mode. In first 3 cases we consider that the circuit was not completed (failure).

- If the leaning angle of the motorcycle is higher than a threshold at some point, the vehicle loses balance and falls (crashes).
- The motorcycle exits the road and does not return to it soon enough.
- The motorcycle makes a turn and returns back to the starting position without completion of the circuit.
- The motorcycle completes the circuit without crashing.

In order to improve on the first crashing condition, I adjusted the crashing threshold, such that the motorcycle crashes at a higher lateral angle. This feature is now more realistic. For the second failure condition I added a timer to calculate for how long the vehicle is off the road. If the vehicle stays off of the road for more than 1 minute then I will reset the motorcycle to its initial position.

In third case the motorcycle will not crash, but it may deviate in and out off the road several times and possibly take a hard turn that takes it backward so that it reaches the starting point again without traversing the entire circuit.. This case can be tested by checking the counters for the centerline points. If the total number of points that the vehicle passed next to is less than the total number of points on the centerline, then we can deduce that the autonomous pilot did not complete the circuit.

In the fourth case the motorcycle completes the circuit. The test for this condition is that the motorcycle reaches the starting point again and all the centerline points show counters with positive values.

The general test will start from a given position on the road ((0, 0, 0) –starting position). The pilot is toggled on with a speed of 0, and then allowed to run until some ending condition is met. At that moment we output and store the statistics. In each trial run of the motorcycle we are interested in following statistics:

- Total time: total time pilot was on (t).
- Average speed and maximum speed (v and max v).
- Total distance covered. This is a measure of how efficiently the pilot has completed the circuit.
- Total number of times the vehicle left the road and average time spent off the road.
- Total number of left and right turns.
- Total number of left and right leans.
- Average lean angle when leaning left or right.
- Average lateral balance: a balance of 0 is achieved when the vehicle is in the center of the road and of 1 when the vehicle is on the border. A lower balance indicates a better road behavior of the pilot.
- Circuit completed or not completed.

4.4 Improvements to the Graphical Application

In the application before the leaning angle of the vehicle was such that even some increase in the leaning angle used to make the motorcycle uncontrollable and the vehicle would go off the road when making a turn in the case where the road is elevated (uphill). I adjusted the leaning angle such that the vehicle can lean to a higher extent and more efficiently.

The crashing angle in the previous setup was such that even if the vehicle was leaning to a small extent, it would crash very often. I adjusted the crash angle such that even if the vehicle leans to a higher extent, it can be made to regain its upright position.

I added some functionality to the motorcycle such that if the vehicle crashes, it can be restarted from the same position. I also attached controls in the human player mode such that if the vehicle crashes outside the road it can be pulled back or pushed front with a small constant speed. This is the equivalent to a manual pull and push of the vehicle in real life.

As I needed to run the motorcycle for a number of trials in order to compare the behavior of the motorcycle under various conditions, I added a functionality to run the motorcycle in the pilot mode for a given number of times. All of the statistics will be stored in a result file.

5. Application of Genetic Algorithms

In this section we introduce the genetic algorithms, the operations, their application, and also an example of GA explained for a particular problem.

5.1 Introduction to Genetic Algorithms

Genetic algorithms (GA) (Holland, 1975; Goldberg 1989) are a part of evolutionary computing, which is a rapidly growing area in the field of artificial intelligence. These algorithms are based on Darwin's theory of evolution. This means that problems are solved by an evolutionary process which is used to optimize the solutions to a given problem (the solution is not always the best). A GA uses a probabilistic process to find approximate solutions to difficult-to-solve problems through application of the principles of evolutionary biology to computer science. Genetic algorithms use biologically-derived techniques such as inheritance, mutation, natural selection, and recombination (or crossover).

Genetic algorithms are typically implemented as a computer simulation in which a population of abstract representations (called chromosomes) of candidate solutions to an optimization problem (called *individuals*) taken from a search space evolves toward better solutions.

Traditionally, potential solutions are represented as binary strings of 0 and 1 values called *genes*. The evolution starts from a population of completely random individuals and takes place in several generations. In each generation, multiple individuals are stochastically selected from the current population, modified (mutated or recombined) to form a new population, which becomes the current population in the next iteration of the algorithm.

A measure of how good a solution is to solve the problem, called *fitness function,* is also necessary in the evolutionary process.

5.2 Comparison of Natural and GA Terminology

The *strings* of artificial genetic systems are analogous to *chromosomes* in biological systems. In natural systems, one or more chromosomes combine to form the total genetic prescription for the construction and operation of some organism. In natural systems the total genetic package is called *genotype*. In artificial genetic systems the total package is called a *structure*. In natural systems the organism formed by the interaction of total genetic package is called the *phenotype*. In artificial genetic systems, the structure decode to form a particular *parameter set, solution alternative, or point* (in the solution space). In natural terminology, we say that chromosomes are composed of genes which may take a number of values called *alleles* and in artificial intelligence we say that strings are composed of *features or detectors*, which take one of the possible values. The position of the gene is called *locus* and in artificial genetic systems we say it is the string *position*.

5.3 Operation of a GA

The algorithm begins with a set of candidate solutions (represented by chromosomes) called population. Potential solutions from one population are taken and used to form a new population. This is motivated by a hope, that the new population will be better than the old one. This assumption was partially explained by the schemata theorem (Goldberg 1989). From the current individuals some are selected to form new potential solutions (offspring). This process uses the fitness of each individual such that the more suitable they are as candidate solutions to the problem, the more chances they have to reproduce.

This process is repeated until some condition (for example achieving a given number of generations or a given improvement of the best potential solution) is satisfied.

Outline of the Basic Genetic Algorithm

1. **[Start]** Generate random population of *n* chromosomes (potential solutions for the problem)

2 **[Loop]** over the following steps until a convergence condition is satisfied.

    1. **[Fitness]** Evaluate the fitness *f(x)* of each chromosome *x* in the population

    2. **[New population]** Create a new population by repeating following steps until the new population is complete

    3. **[Selection]** Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)

    4. **[Crossover]** With a crossover probability cross over the parents to form new ones (offspring or children). If no crossover was performed, the offspring is the exact copy of the parents.

5. **[Mutation]** With a mutation probability, mutate the new offspring at each locus (position in chromosome).

6. **[Accepting]** Place the new offspring in the new population.

7. **[Replace]** Use new generated population for a further run of the algorithm.

8. **[Test]** If the end condition is satisfied, **stop**, and return the best solution in current population

3 **[Return]** the best solution in the last generation.

5.4 Genetic Algorithm Explained with an Example

Initial Population

A genetic algorithm starts with a population of strings to be able to generate successive populations of strings afterwards. The initialization is done randomly. This means to say that every gene is set to 0 or 1, with each value having a chance of 50% to occur.

In our problem we have chosen each of the parameters with a 10 bits representation. Each of them has the inferior limit of 0 and maximum of 10. In order to explain this better, let us consider an example. Suppose we need to find the maximum of the function

$$u(x, y) = (x - 7)^2 + (y - 3)^2 \tag{29}$$

with both x and y taking values in an interval range of [0,7].

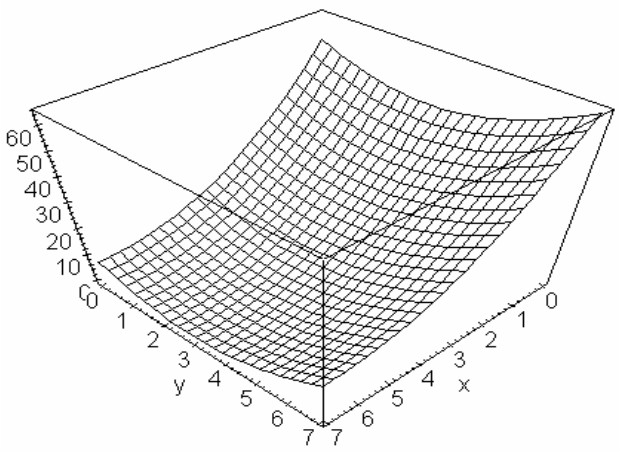

Figure 7. Graph of the function $u(x,y) = (x-7)^2 + (y-3)^2$

We need to generate a code for this problem. A binary representation of six bits has been chosen, where the three bits on the left represent x and the three bits on right represent y.

Let us assume that the following table represents the initial population of strings selected by successive flip of coins.

Table 1. Initial population

Number	String	Values
1	100001	(x = 4, y =1)
2	001100	(x= 1, y = 4)
3	110010	(x =6, y = 4)
4	000100	(x = 0, y = 4)

Evaluation

After every generated population, every individual in the population must be evaluated so that we can select the better ones. This is done by comparing the individuals with the *fitness function.* In this case we consider f(x) = u(x). By substituting the values of x and y in the equation we get the fitness of 13, 37, 2, 50 for the initial population.

Reproduction

It is very important to decide which individuals will be chosen for the purpose of procreation. In GA this selection is based on the string fitness. According to the 'survival of fittest' principle, if a string A is twice as fit as string B, then A is expected to appear twice as much in the next generation. This kind of implementation of the reproduction is done by creating a biased roulette wheel where each current string in the population has a roulette wheel slot sized in proportion to its fitness [Goldberg, 1989]. By dividing the individual fitness by the average of all fitness values, we can calculate the expected count of this individual in next generation. In the example we explain, the average fitness in 25.5. So the expected count of individual one in the next generation is 13/25.5. = 0.52. Other expected counts are shown in table 2, as well as the normalized fitness values, which are equal to the fitness values divided by the total sum of all fitness values (102 in our example), multiplied by 100%. The normalized fitness gives the chance of an individual to be chosen as a parent. A method to actually select an individual as a parent is to use a sum function $S_i = ?^{i}_{j=1} f_i$, (the sum of all fitness values from individual one to individual i), and randomly and uniformly choose an integer between 0 and the sum of all fitness values. The first individual whose S_i is equal or greater than this integer will be chosen as a parent. The S_i values are shown in table 3. For example, suppose that the randomly chosen number is 53, then the individual 4 will be chosen as a parent because S_4 is the first value that succeeds 53. This routine will be repeated until we have 4 parents.

The parent is as shown in Table 2 below.

Table 2. Reproduction results

Number	String	(x,y)	Fitness	Normalized	Si	Expected count	Actual
1	100001	(4,1)	13	12.7%	13	0.51	1
2	001100	(1,4)	37	36.3%	50	1.45	1
3	110010	(6,2)	2	0.20%	52	0.08	0
4	000100	(0,4)	50	49.0%	102	1.96	2

Crossover

Once the two parents are selected from the previous step, the genetic algorithm combines them to create two new offspring. Combination is done by the crossover operator. We have selected the one-point crossover for our experiments.

One-point crossover

A random crossover point is selected. The first part of the first parent is combined with the second part of the second parent to make the first offspring. The second offspring will be built from the second part of the first parent and first part of the second one.

For the example we have chosen above, the random crossover point is selected between the last two genes.

Parent #1: 10000 | 1

Parent#2: 11001 | 0

The resulting offspring is as following:

Offspring #1 : 100000

Offspring #2: 110011

One of the most important aspects of crossover is that one-point crossover cannot generate certain combinations of features encoded on chromosomes: schemata with a large defining length are easily disrupted. It is also possible that certain elements are not allowed to appear more than once. In that case, precautions have to be taken.

Two-Point crossover

The two-point crossover operator differs from the one point crossover in the fact that two crossover points are selected for the operation. Starting from the same parents as above, let us suppose that the crossover points are chosen as shown below:

Parent #1: 100 | 00| 1

Parent#2: 110 | 01| 0

The offspring in this case will be the following:

Offspring #1: 100 | 01 | 0

Offspring #2: 110 | 00 | 1

Uniform crossover

In the uniform crossover each gene is selected randomly, whether from the first part parent or from the second one, with a certain probability.

Parent #1: 100001

Parent#2: 110010

The offspring in this case is as follows

Offspring #1 : 1**1**000**0**

Offspring #2: 1**0**001**1**

Mutation

The mutation is the genetic operator that randomly changes one or more of the chromosome's genes. The purpose of the mutation operator is to prevent the genetic population from converging to a local minimum and to introduce in the population new possible solutions. The mutation is carried out according to the mutation probability.

The mutating operator simply tosses a biased coin with probability p_{mutate}(which is very small) at each bit and, according to that result, changes a 1 into a 0 and vice versa.

Table 3 shows the result of the mutation operator.

Table 3. Mutation operation

Before Mutation	**After Mutation**
10**1**100	10**0**100

Table 4. New population and fitness after crossover and mutation

Number	Selected parents	After crossover	After mutation	New fitness
1	10|0001	101100	10**0**100	10
2	00|1100	000001	000001	53
3	00010|0	000100	000100	50
4	00010|0	000100	000100	50

As we expected, a new string with high fitness has appeared. The sum of the fitness in the population increased from 102 to 163 and the average has also increased from 25.5 to 40.8. In the initial population, strings 1 and 2 were selected (average fitness), string 3 was not selected (low fitness) and string 4 was selected twice due to its high fitness. Crossover helped us by providing the high fitness string 000001(string2) and also string1 which has a lower fitness of 10.

This process of reproduction crossover and mutation is carried out until there is no further change in terms of fitness. Then we say that we have reached the optimal fitness value for the given problem.

5.5 Applications of Genetic Algorithms

GAs are widely used in various types of problem solving, for modeling, and also in various scientific, and engineering problems.

Following are some of the applications of GA

- **Optimization**: GAs are used in a wide variety of optimization tasks like numerical optimization, and combinational optimization problems such as travelling salesman problem(TSP), circuit design, job shop scheduling, and video and sound quality optimization.
- **Automatic Programming**: used to evolve computer programs for specific tasks, and to design other computational structures, for example, cellular automata and sorting networks.
- **Machine and robot learning**: used for many machine- learning applications, including classification and prediction, and protein structure prediction. GAs have also been used to design neural networks, to evolve rules for learning classifier systems or symbolic production systems, and to design and control robots.
- **Economic models**: used to model processes of innovation, the development of bidding strategies, and the emergence of economic markets.
- **Immune system models**: used to model various aspects of the natural immune system, including somatic mutation during an individual's lifetime and the discovery of multi-gene families during evolutionary time.
- **Ecological models**: used to model ecological phenomena such as biological arms races, host-parasite co-evolutions, symbiosis and resource flow in ecologies.

5.6 Genetic Representation of the Motorcycle Pilot

To apply the GAs to our problem we need to find a good representation of the potential solutions as chromosomes and to find a good fitness function. The behavior of the agents is described by some equations. For example, in Section 3.4 we have introduced the throttle agent. The genetic representation of this agent starts with the sequence of all configurable parameters occurring in equations (17) to (22).

$S = (lat_{norm}, leftd, rightd, v(t), v_{low}, lat_{abs}, thr_{lat}, front, thr_{front}, vr_{limit}, c_{tr}, tr(t), tr_{lat}, pvlat, tr_{fr}, c_{vfr}, pvfr, c_{incv}, c_{dec}, c_{sl})$

These are all real numbers that we can further represent as a sequence of binary genes (for example 10 genes for each parameter). For example, if the $0 = thr_{front} = 10$, then the sequence 00….0 will represent $thr_{front} = 0$, 11…1 will represent $thr_{front} = 10$, 10…0 for $thr_{front} = 5$, and so on. This idea is shown in Equation 30.

$$\begin{cases} real(0L, lower, upper) = real\left(L, lower, \dfrac{upper + lower}{2}\right) \\ real(1L, lower, upper) = real\left(L, \dfrac{upper + lower}{2}, upper\right) \\ real(0, lower, upper) = lower \\ real(1, lower, upper) = upper \end{cases} \qquad (30)$$

where L is any sequence of bits. In our case we start with lower = 0 and upper = 10. More precisely, the sequence 1000110111 can be converted as:

real(1000110111) = 1(5) + 0(2.5) + 0(1.25) + 0(0.625) + 1(0.3125) + 1(0.15625) + 0(0.078125)

+ 1(0.0390625) + 1(0.0195312) + 1(0.0097656) =

= ~ 5.5371093

The final chromosome is a concatenation of all these sequences of genes for all of the parameters. There are 32 parameters total for our pilot and so the chromosomes have a length of 320.

The fitness function will be computed by running the motorcycle in a non-graphical environment over the entire circuit using the real representation of the chromosome as configuration for the autonomous pilot. The criteria for computing the fitness will involve how far the pilot went on the circuit before crashing or getting out of the road, and how fast it completed the circuit in case of success.

At the end of experiments, the parameters derived by the GAs can be imported back into the graphical application for a visual verification of the quality of the solution and for testing the pilot with these new parameters under the same conditions as we have done for the manually configured one.

Fitness Function

The fitness function to evaluate the quality of the pilot represented by a chromosome was defined as below

Let

$F(x)$ = Fitness Function

d_m = number of points touched by the motorcycle

d_t = total number of points

t_m = total time taken by the motor cycle before crashing or completing the circuit

Each time this fitness value is noted and at the end of number of generations

$$F(x) = \begin{cases} \frac{d_m}{d_t} + \frac{1}{1 + t_m} & \text{if the circuit was completed} \\ \frac{d_m}{d_t} + \frac{1}{5 + t_m} & \text{if the circuit was not completed} \end{cases} \qquad (31)$$

The second part of the Equation 31 is designed to put higher accent on the percentage of the circuit completed by the pilot than on the time when circuit was not completed.

Implementation Details

Figures 8, 9, 10, 11shows the interface to the genetic algorithm with the settings we have used for our experiments.

Figure 8. Main application screen GA settings

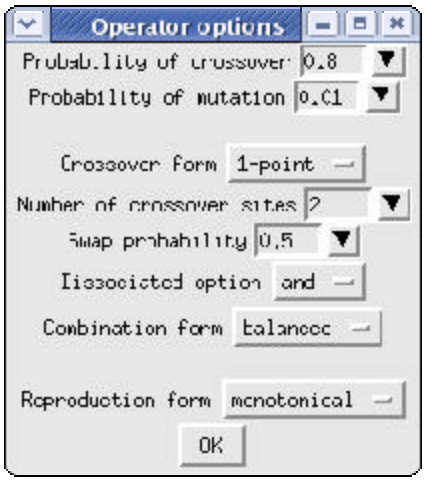

Figure 9. Running options settings

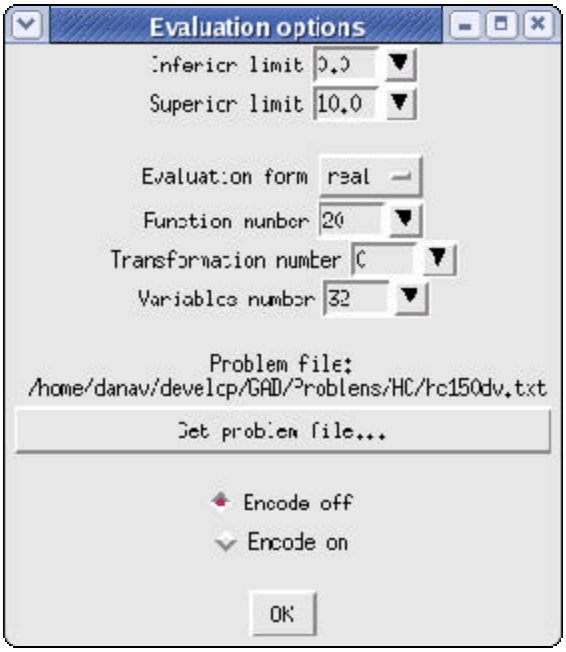

Figure 10. Evaluation options settings

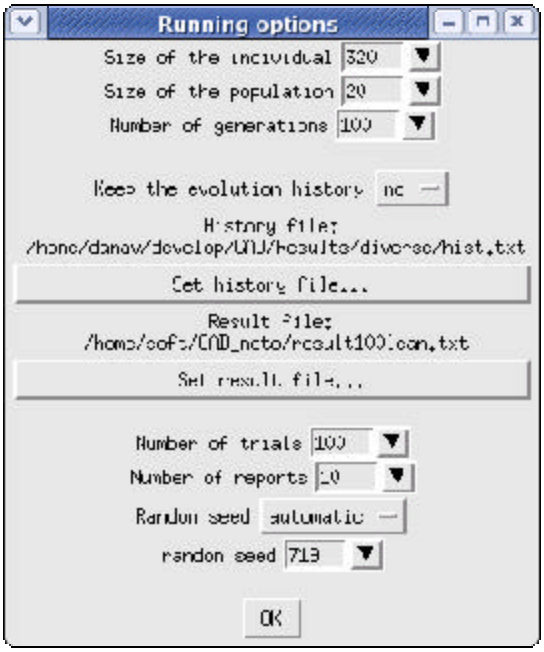

Figure 11. Running options settings

6. Comparison Study

Table 5 shows the statistics for two human players reported in (Vrajitoru, Mehler, 2004). We include these results so that they can be compared with the autonomous pilot.

Table 5.Statistics for two human players

Measure	Steering	
	Human1	**Human2**
Total time	97.4	79.2
Total distance	2312.05	2316.83
Speed	6.19	8.94
Max speed	8.75	12.26
Lateral balance	0.29	0.36
Left turns	121.4	119.2
Right turns	51.4	47
Times it left the road	0	0.4
Recovery Time	0	11.2
Number of circuits	100%	100%
Perfect circuits	100%	60%

In order to compare the behavior of the motorcycle in case of using genetic algorithm with the heuristic approach, Table 6 show the results obtained in steering and leaning mode for 100 trials before applying the genetic algorithms.

From Table 6 we can see that the autonomous pilot was capable of completing the circuit 92% of the time in approximately 6.5 minutes on the average. The pilot did not complete the circuit in the leaning mode.

The average completion time for the human players shown in Table 5 was approximately 1.5 minutes, so the autonomous pilot configured manually showed a fairly lower performance than the human players.

Table 6. Average result of 100 trails

Measure	Steering		Leaning
	Completed circuits	Incomplete circuits	Incomplete circuits
Total time	327.75	78.25	15.79
Total distance	2338.77	720.421	77.2304
Speed	1.95824	2.58261	1.24538
Max speed	4.98943	4.95509	2.5573
Lateral balance	0.323663	0.327008	0.547597
Left turns	128.457	45.75	0
Right turns	58.3804	22.25	0
Left leans	0	0	4.53
Right leans	0	0	7.74
Leaning angle to the left	0	0	-7.60397
Frames leaning left	0	0	105.1
Leaning angle to the right	0	0	9.00594
Frames leaning right	0	0	25.16
Times it left the road	0.347826	1.75	0.4
Frames spent out of the road	2.30435	8.625	15.545
Number of circuits	92	8	100

The Table 7 shows the average fitness function for both leaning mode and steering mode which is obtained by GA.

Table 7. Average fitness in 100 trails for 100 generations

Generation	Steer	Lean
0	1.76074	0.273014
10	1.80534	0.311289
20	1.84572	0.327876
30	1.85608	0.33766
40	1.86285	0.344841
50	1.86695	0.35315
60	1.88226	0.358102
70	1.8854	0.363652
80	1.90286	0.369676
90	1.90404	0.379083
100	1.90845	0.380829

Figure 11 shows the graph of the average fitness evolution in the steering and leaning mode during the execution of the GA. This shows the best average fitness achieved in 100 generations is around 1.9 in the steer mode and the best fitness is 0.38 in lean mode.

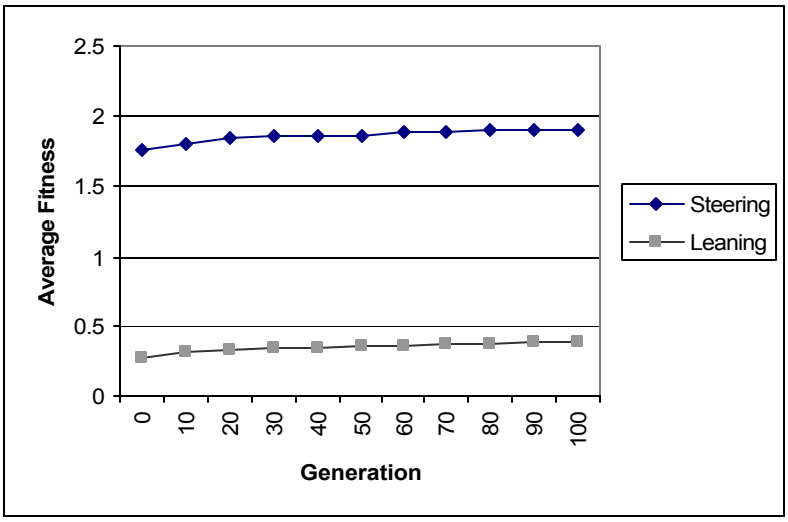

Figure 12. Average fitness in steering and leaning mode with the GA

All the trial runs are done in the non GUI mode for GA parameter calculation. So in order to visually verify the obtained results, I imported the parameters derived by the best GA run back into the GUI application of the motorcycle and did a run in steering and leaning mode for 100 trials. The results obtained are as in Table 8 below.

From Table 6 and Table 8 it is clearly evident that the time taken by the motorcycle is reduced by 60%, the speed of the motorcycle has increased by more than 100% in steering mode with parameters obtained by the GA. Also we can observe that in steering mode in Table 6 there are 6 incompleted circuits in steer mode, but with the parameters derived by the GA, all of the 100 circuits are completed in steer mode.

There is also one more interesting result in the behavior of the motorcycle in lean mode that can be observed after applying the GA. In Table 6 as we see there are no completed circuits in lean mode whereas by applying the parameters obtained from the GA, we observe there are one circuit completed the by motorcycle. The average time that the morotcycle spends on the road before crashing in case of the incomplete circuits has also increased from 15.79s to 52.6s which is 333.12%.

Table 8. Verification of GA parameters in motorcycle GUI application

Measure	Steering	Leaning	
	Completed circuits	Completed circuits	Incomplete circuits
Total time	133.73	221	52.596
Total distance	2333.33	2356.08	522.457
Speed	4.16	2.7354	2.07084
Max speed	6.48	6.07947	5.08668
Lateral balance	0.4	0.396827	0.463341
Left turns	112.04	0	0
Right turns	113.51	0	0
Left leans	0	5	4.79798
Right leans	0	4	4.63636
Leaning angle to the left	0	-3	-4.00343
Frames leaning left	0	2724	633.121
Leaning angle to the right	0	3	4.29006
Frames leaning right	0	1444	285.253
Times it left the road	0.54	0	0.020202
Frames spent out of the road	2.89	0	1.0101
Number of circuits	100	1	99

7. Conclusions

In this thesis it has been shown that genetic algorithm performed better in choosing the coefficients that determine the behavior of the pilot than the manual configuration of the pilot chosen by the user in previous application. The experiments in Section 6 have shown that the autonomous pilot is capable of successfully driving the motorcycle over the entire length of a test circuit in conditions that are comparable to human driver. The time taken by the driver to complete a circuit using GAs is less than 50% of the time compared to the previous settings of the pilot. We observe that the number of left and right turns taken by the driver in the case of GA is lot lower which indicates a higher performance in choosing the correct path on the road. In the previous application the driver was more sensitive to the differences in left and right distances than a human player and the general impression of the ride was less smooth. Using GA's the motorcycle ride is smoother and the simulation is closer to a real life situation.

Our experiments have set the premises for a more complete and thorough evaluation of the autonomous pilot and have proven that the genetic algorithms represent a valid approach for successfully and efficiently configuring the autonomous pilot.

The application of GA presented in this thesis can be extended to other types of vehicles or to autonomous robots.

8. References

T. Al-Shihabi and R.R. Mourant (2003): **Toward more realistic behavior models for autonomous vehicles in driving simulators**. *Transportation Research Record*, (1843):41.49, 2003.

T. F. Abdelzaher, E. M. Atkins, and K. G. Shin (2000): **QoS negotiation in real-time systems and its application to automated flight control**. *IEEE Transactions on Computers,* 49(11):1170.1183, Best of RTAS '97 Special Issue.

E. M. Atkins, R. H. Miller, T. VanPelt, K. D. Shaw,W. B. Ribbens, P. D. Washabaugh, and D. S. Bernstein (1998): **Solus: An autonomous aircraft for flight control and trajectory planning research**. In *Proceedings of the American Control Conference (ACC*), volume 2, pages 689.693.

Andreas Fuchs(2000): **Trim of aerodynamically faired single-track vehicles in crosswinds.** *Proceedings of the 3rd European-Seminar on Velomobiles,* Roskilde, Denmark.

J. Fajans (2000): **Steering in bicycles and motorcycles** The *American Journal of Physics* 68, p 654.

V. Gavrilets, E. Frazzoli, B. Mettler, M. Piedmonte, and E. Feron (2001): **Aggressive maneuvering of small helicopters**: **a human centered approach**. *International Journal on Robotics Research.*

Goldberg, D. E. (1989): *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading (MA): Addison-Wesley.

Holland, J. H. (1975): *Adaptation in Natural and Artificial Systems*. Ann Arbor: University of Michigan Press.

Kamran H. Sedighi, Kaveh Ashenayi, Theodore W. Manikas,Roger L. Wainwright, Heng-Ming Tai (2004)**:** *Autonomous Local Path-Planning for a Mobile Robot Using a Genetic Algorithm***,** Technical report, Electrical Engineering and Computer Science Dept.University of Tulsa, Tulsa, OK 74104.

T. Kelly (1997): **Driver strategy and traffic system performance**. *Physica A*, 235:407.417.

J. De Mot and E. Feron (2003): **Spatial distribution of two-agent clusters for efficient navigation**. *In IEEE Conference on Decision and Control,* Maui, HI.

R.R. Mourant and S. Marangos (2003): **A virtual environments editor for driving scenes**. In *Proceedings of the International Conference on Computer, Communication and Control Technologies,* volume IV, pages 379.384, Orlando, Florida.

M. Mohan, D. Busquets, R.López de Màntaras and C. Sierra (2004): **Integrating a potential field based pilot into a multiagent navigation architecture for autonomous robots**. *Proceedings of ICINCO'04 (Vol. 2), pp. 287-290, INSTICC Press.*

K. Nagel (1996): **Particle hopping models and traffic flow theory**. *Physical Review E*, 53:4655.

J. Olsen, J.M. Papadopoulos (1988): **Bicycle dynamics (The meaning behind math).** *Bike Tech.*

Rahul Sukthankar, Shumeet Baluja, and John Hancock (1998): **Multiple adaptive agents for tactical driving**. *Applied Intelligence*, 9(1):7.23.

Robin Biesbroek (2001): *Genetic Algorithms Tutorial*:
http://www.estec.esa.nl/outreach/team/robin_biesbroek.htm

D. Vrajitoru, R. Mehler (2004): **Multi-agent autonomous pilot for motorcycles**.: The *IEEE Region 4 Electro/Information Technology Conference (EIT2004)*, August 26-27, Milwaukee, WI.